2007-03-07

LUNDS TEKNISKA HÖGSKOLA
Lunds universitet

Department of
Information Technology

Rapport i Digitala Projekt 5p

Temperaturregulator

Grupp 16; Anders Sandberg M03
Bengt Westman M03

Abstract

The goal with this project was to develop a device that reads values from temperature sensors and according to the readings actuates a thermostat. Furthermore the data should be kept in a log with the accurate date and time tagged to the readings. The platform was the Motorola 68008 processor. The handling of this processor compared to a single-chip processor turned out to be quite complex. As a consequence of that a substantial part of the time was spent on making the processor communicate, e.g. handle the signals, in the right way instead of the functionality in the finished device. Though the quantity measured in this case is temperature, it must be elucidated that the solution is very general and various kinds of sensors could have been used instead.

Innehållsförteckning

Inledning

Att med hjälp av sensorer mäta en fysikalisk storhet och på grundval av dessa mätvärden utföra en reglering är målet som ska nås med denna konstruktion. Det är dock så att i detta projekt är det vägen dit som är av intresse och inte funktionaliteten i den färdiga konstruktionen. Processorn Motorola 68008 är plattformen för konstruktionen. Att jämföra denna med en enchipsprocessor låter sig lättast göras med en analogi. Säg att målet är köra mellan Lund och Malmö och verktyget för att utföra detta är en bil. Med enchipsprocessorn skulle detta innebära att en nästan färdig bil finns tillgänglig. Det enda som behöver göras är att tanka, sätta på hjul, ställa in färddatorn och köra. I Motorolans fall är utgångspunkten enbart motorn. Att avgöra vad som mer behövs för att färdigställa bilen och uppnå målet är upp till var och en. Arbetet inleds med att sammanställa en principskiss över konstruktionen. När denna är fastställd kan de olika periferienheterna dimensioneras. IT-instutionen tillhandahåller en databank med komponenter som är kompatibla med processorn. Från denna väljs matchande komponenter ut. Nu inleds byggfasen. Komponenterna viras fast på ett prototyp kort. Testning av hårdvara och mjukvara utförs med It-68 emulatorn. Denna modul simulerar processor och EPROM. Via ett gränssnitt med en Pc kan konstruktionen testas och felsökas. Hårdvaran testas först. Varje enhet testas för sig och endast enkla test utförs. Vissa färdiga testrutiner finns inbyggda men generellt sett testas hårdvaran genom att specifika adresser tilldelas ett värde och sedan avläses. Med hjälp av en logikpenna kontrolleras att rätt signaler genereras vid rätt tidpunkt. När det är fastställt att processorn kan kommunicera med övrig hårdvara kan mjukvaran skrivas. Rutiner för olika funktioner skrivs och successivt testas funktionaliteten. Detta leder till att kontaktfel, felkopplingar etc. uppdagas och åtgärdas. Vissa kompletteringar av hårdvaran krävs när nya insikter nås. Denna rapport beskriver i stort denna process fram till färdig hårdvara med önskvärd funktionalitet.

Metod och genomförande

Kravspecifikation

Konstruktionen har inga större krav på snabbhet då det är temperatur som ska mätas. Temperatur ändras sakta i förhållande till de tider som en mikroprocessor arbetar med. Två temperaturer ska mätas, inomhus och utomhus temperatur. Dessa ska lagras undan tillsammans med tiden då de uppmättes. Data ska på operatörens kommando överföras och lagras på en PC. Vid dimensionering av ramminne har antagits att data överförs till PC en gång om dagen. På grundval av avlästa temperaturer ska en åtgärd kunna utföras via D/A-omvandlaren, i detta fall ställa in en termostat.

Teori

För att få fördjupad förståelse för hur en dator fungerar väljs inte en enchipsprocessor där stora delar av datorn är inbyggd. Istället väljs en fristående processor där periferienheter väljs efter behov. För att en processor ska kunna utföra instruktioner krävs ett minne där programkoden ligger. Detta minne ska inte tömmas om strömmen bryts utan det ska bibehållas oavsett strömförsörjning. För detta krävs därför ett PROM minne. Då processorn utför instruktioner i programkoden används diverse variabler. Dessa behöver också sparas undan, men då i ett minne som lätt kan skrivas över och återanvändas. För detta ändamål behövs ett RAM minne. För att processorn ska kunna styra applikationen krävs slutligen att en eller flera enklare logikkretsar ansluts, så kallade PLD-kretsar. Dessa kan utföra enkla kombinatoriska operationer så att processorns signaler kan tolkas och rätt signaler skickas vidare till kringutrustningen. Nu fungerar datorn men kan inte utföra något meningsfullt. För att kunna utföra något nyttigt med signalerna behövs olika periferienheter t.ex. A/D-omvandlare, D/A-omvandlare, UART, RT-klocka m.m. Vilka som väljs beror på applikationen.

Val av komponenter

Med hänsyn till kravspecifikationen väljs en processor från Motorola, Motorola 68008. Processorn har en stor minneskapacitet i förhållande till många enchipsprocessorer vilket ger möjlighet till att kombinera minnesstorlekar och kringenheter efter behov. Processorn ger en bra bas för vidare utveckling och anslutning av extra kringenheter då den kan adressera en stor minnesarea. Till processorn väljs en 8 MHz klocka.

Konstruktionen behöver ett PROM minne för att lagra programkod. Den kod som krävs är mycket begränsad varför ett litet vanligt förekommande minne väljs. Minnet som väljs är ett EPROM av National Semiconducters, 27C64. Minnet har en kapacitet på 8k.

För lagring av variabler och stack används ett SRAM från NEC. Alla de värden som ska sparas undan kommer att läggas i SRAM minnet varför ett större minne väljs. Minnet som väljs är 43256, och har en kapacitet på 32k.

De sensorer som används för temperaturmätning är analoga vilket medför att en A/D-omvandlare behövs. Eftersom flera sensorer ska läsas av väljs en A/D-omvandlare med flera kanaler istället för att använda flera A/D-omvandlare. Den A/D-omvandlaren som väljs är ADC0809 från National Semiconducters, vilket är en åtta

bitars omvandlare. Omvandlaren har åtta stycken kanaler vilket ger en viss överkapacitet som kan användas för eventuell framtida utbyggnad.

En termostat som ska styras kan ha en analog styrsignal eller ha en on-off funktion. För att vara oberoende termostat ansluts en D/A-omvandlare till kortet. Den D/A omvandlare som väljs är TLC7524 som är en åtta bitars omvandlare från Texas Instruments. Då det endast är en termostat som ska styras väljs ingen med flera kanaler.

Mätningar ska ske med jämna mellanrum med en tidsangivelse vid varje mätvärde. För detta behövs en realtidsklocka. Klockan behöver kunna generera avbrott med varierande tidssteg. Den klocka som väljs är ICM7170 från Harris semiconducter, vilket är en 8 bitars klocka. Klockan behöver en kristall med bestämd frekvens för att kunna fungera korrekt. Efter kontroll i datablad väljs en 37.768kHz kristall.

För att kunna logga data i en PC behövs en kommunikationsväg till PC'n. På PC'n sitter en COM-port som kan användas. För att kunna kommunicera med den behövs en UART. Den UART som väljs är från EXAR, XR68C681. Det är en tvåkanalig seriell asynkron mottagare/sändare. För att denna ska kunna kommunicera med datorn enligt RS232 standarden behövs även en driver. Driverns uppgift är att förstärka signalen från UART'en till ±12V och försvaga signalen från RS232 så att den passar TTL standard på 0-5V. Den driver som väljs är MAX233 från MAXIM. UART'en behöver även en kristall för att kunna skicka data med rätt hastighet. Denna kristall väljs efter databladsspecifikationer till en 3.6864MHz kristall, detta för att med hjälp av den interna bitrate-generatorn kunna producera förutbestämda standardiserade bit-rates.

För val av enheter, styrning av avbrottssignaler m.m. behövs logik-kretsar. Efter en sammanfattning av de signaler som ska styras konstateras att två stycken PALCE22V10 behövs. Dessa har tolv ingångar och tio in/utgångar.

Utöver dessa enheter används en RESET-knapp, motstånd och kapacitanser. Motstånden används som pull-up-motstånd. Kapacitanserna används dels för att stabilisera och undvika uppstartningsproblem med kristallerna och dels för att filtrera bort störningar på spänningsförsörjningen.

Konstruktion

Konstruktionen bygger på ett färdigt prototypkort med skenor för Vcc och jord. Komponenterna monteras och fästs när lödning av Vcc och jord sker. Det är bara spänningsförsörjningen som löds, för kontakt mellan olika pinnar används ett virningsverktyg som virar tråden runt pinnen. Pinnarna är fyrkantiga och vid virningen dras tråden så hårt runt pinnen att den sträcks och därmed får en bra kontaktyta och sitter hårt. På detta sätt görs alla kontakter mellan de olika enheterna. Databuss och adressbuss går från enhet till enhet för att minska antalet kontakter.

Felsökning

Felsökningen består av flera delar. Innan konstruktionen kan användas och programmering kan börja ske måste all hårdvara fungera. För att testa hårdvaran används utvecklingssystemet it-68, vilket kopplas mellan en PC och konstruktionen. Med en hyperterminal kan kommunikation ske som om PC'n vore processorn.

Felsökning och testning av hårdvaran utgörs främst av förmåga att kunna skriva till och läsa olika register och adresser. Om kommunikation med en enhet fungerar testas nästa. Om en enhet inte beter sig som förväntat används en logikpenna för att följa de signaler som ska genereras.

När hårdvaran är testad kan programmering börja. Efter att ett program är skrivet kan det testas genom att ladda ner det till it-68. Om programmet inte fungerar som förväntat kan brytpunkter användas för att se hur långt programmet fungerar. Nästa steg är att använda logikpennan för kontroll av alla signaler som bör genereras.

Programmering

Programmering till processorn sker mestadels i C. Vissa avbrottshanteringsrutiner, initiering m.m. finns färdigt i en mall av institutionen för informations teknologi. C är ett hårdvarunära språk vilket möjliggör för användning av pekare. Pekarna används för adressering av register i de olika kringenheterna. Programmet består av deklaration av variabler, pekare och vektorer, initieringsrutiner för RT-klocka och UART, ett Main-program och avbrottshanteringsrutiner.

Resultat

Projektet har nått goda resultat i form av ökad kunskap inom digitalteknik. Funktionaliteten hos konstruktionen är inte så hög som önskad. Alla de enheter som har valts för konstruktionen fungerar. De mål som inte nåtts är att grafiskt visa mätvärden i diagram. Problemet ligger dock inte inom digitalteknik, utan mer inom programmering. En stor del av projektet har bestått av felsökning för att få konstruktionen att fungera vilket tyvärr inte syns i det färdiga resultatet.

Projektet har efter tidens gång resulterat i en lista över möjliga fel vid felsökning förutom de som ges av it-68.

Benämning/Symptom	Åtgärd
RAM test ger felmeddelande	Kontrollera ICE-kabeln Kontrollera virningar, skalad kabel får inte ligga fri.
Oregelbundna felmeddelanden	Samma som ovan
Fel värde på databus	En skalad kabel kan ligga åt nästa pinne vilket ger samma värden på dessa
Konstant avbrott	Fel logik i PLD
Ingen utsignal från UART	Fel initiering av UART

Diskussion

Under utvecklingen av denna konstruktion har diverse problem uppstått. Felsökning och åtgärder har stått för en stor del av arbetet. Felen har främst legat i konstruktionen men även mjukvaran och brott på flatkabeln från it-68 emulatorn har ställt till problem. När flera fel uppstår i kombination och dessutom oregelbundet blir felsökningen mycket svår. För att få bukt med problemen är det effektivast att starta om felsökningen från början genom att avlägsna alla enheter som inte behövs. Genom att göra detta kan felen lättare hittas. Vid virning av kablar är det viktigt att se till att inte kabeln inte skalad för långt så att kontakt uppstår med andra pinnar.

Referenser

www-adresser;

Datablad processor. 2007-02-27
http://www.it.lth.se/digp/datablad/datablad_index.asp

Datablad EPROM. 2007-02-27
http://www.it.lth.se/datablad/Memory/eprom/27c64.pdf

Datablad SRAM. 2007-02-27
http://www.it.lth.se/datablad/Memory/sram/43256.pdf

Datablad PLD. 2007-02-27
http://www.it.lth.se/datablad/logik/Programmable/Palce22v10.pdf

Datablad UART. 2007-02-27
http://www.it.lth.se/datablad/periphery/communication/xr68c681.pdf

Datablad driver. 2007-02-27
http://www.it.lth.se/datablad/periphery/communication/max232-233.pdf

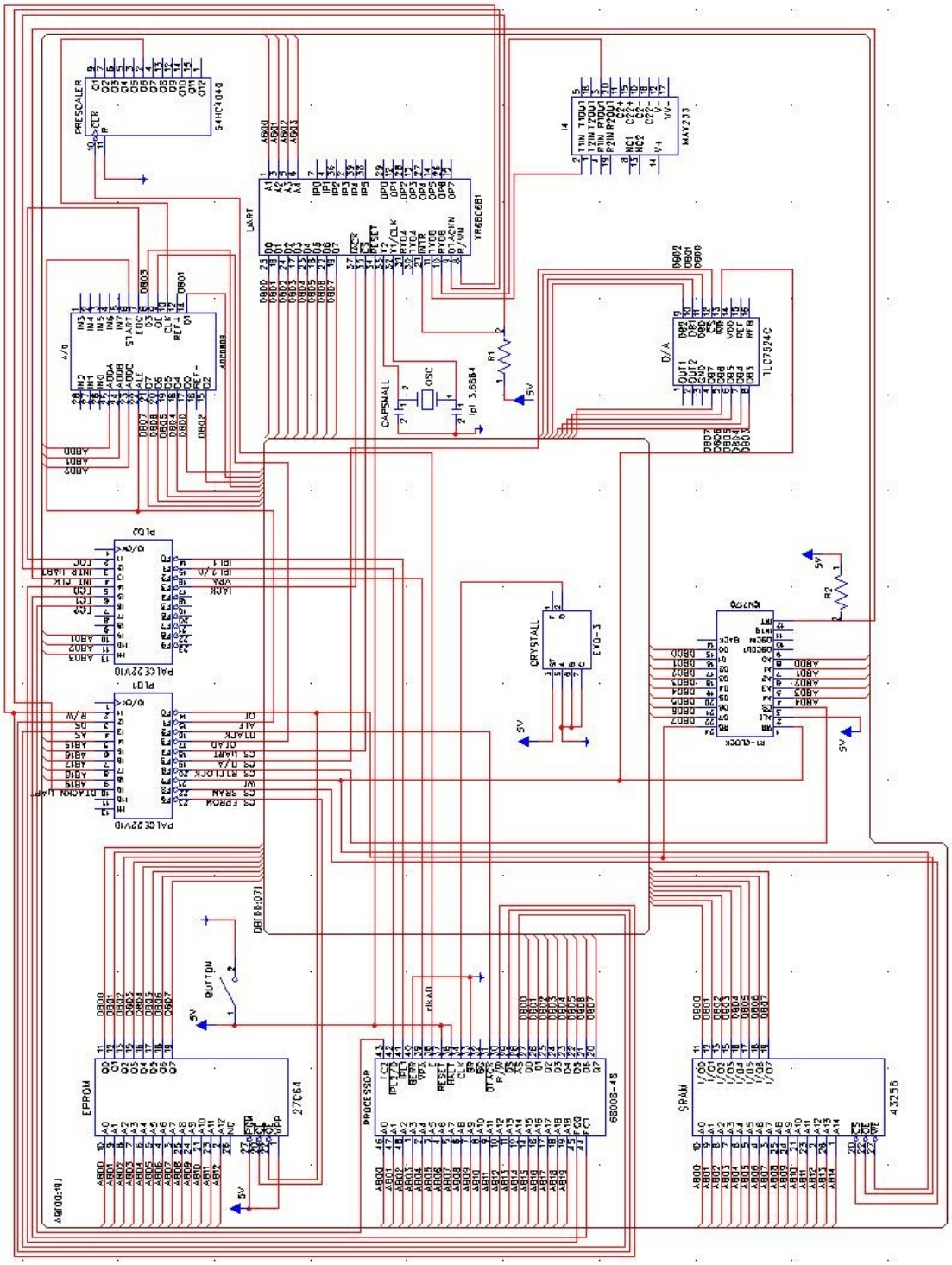
Datablad A/D-omvandlare. 2007-02-27
http://www.it.lth.se/datablad/Analog/adda/adc0809.pdf

Datablad D/A-omvandlare. 2007-02-27
http://www.it.lth.se/datablad/Analog/adda/tlc7524.pdf

it-68 utvecklingssystem. 2007-02-27
http://www.it.lth.se/digp/PDF_files/it68/it68.pdf

Bilagor

Ritning



PLD-kod
PLD1-kod

```
device 22V10

RW     2       'Read write signal från processor. Fungerar ihop med DS
DS     3       'Indikerar att det finns något att läsa på databussen
AS     4       'Indikerar att det finns något att läsa på adressbussen
AB15   5       'Adressbuss
AB16   6
AB17   7
AB18   8
AB19   9
DTACKu         10
GND    12
OE     14      'Fungerar som lässignal
ALE    15      'Indikerar att enheten(AD) ska läsa analog signal. Man väljer även
ingång 1-8
DTACK          16      'Indikerar att dataöverföring är klar
OEAD   17
CSUART         18      'Indikerar att denna enhet ska vara aktiv
CSDA   19      'Indikerar att vi ska läsa omv. värde på AD
CSRTC          20
WE     21      'Anger att data kan läggas ut på bussen
CSSRAM         22
CSEPROM        23
VCC    24

start

OE/=/DS*RW;
ALE=AB15*/AB16*AB17*/AB18*/AB19*/RW*/AS;
OEAD=AB15*/AB16*AB17*/AB18*/AB19*RW*/DS;
CSUART/=/AB15*/AB16*AB17*/AB18*/AB19*/AS;
CSDA/=/AB15*AB16*AB17*/AB18*/AB19*/AS;
CSRTC/=AB15*AB16*/AB17*/AB18*/AB19*/AS;
WE/=/DS*/RW;
CSSRAM/=AB15*/AB16*/AB17*/AB18*/AB19*/AS;
CSEPROM/=/AB15*/AB16*/AB17*/AB18*/AB19*/AS;
DTACK/=/CSDA+/CSRTC+/CSSRAM+/CSEPROM+ALE+/DTACKu+OEAD;

End
```

PLD2-kod

```
device 22V10

EOC    2
INTRUART   3
INTRTC     4
FC0    5
FC1    6
FC2    7
AB1    10
AB2    11
GND    12
AB3    13
IPL1   14
IPL20  15
VPA    16
IACK   17
VCC    24

start

IPL1/=EOC*INTRUART+/INTRTC*INTRUART;
IPL20/=/INTRUART;
VPA/=FC0*FC1*FC2*/AB1*AB2*/AB3;
IACK/=FC0*FC1*FC2*AB1*/AB2*AB3;

end
```

Programkod

```
/*****************************************************************
*****
                            test.c
                            ------
   Detta ar huvudprogrammet som ar skrivet i ANSI C. Exekveringen av
hela
   programpaketet borjar i pmain.68k (lage __main).

   exp4() anropas fran assemblyprogrammet exp4.68k vid avbrott.

   _avben() anropar avben.68k vilket tillater avbrott fran PI/T.
*****************************************************************
*******/
/*Definierar pekare och variabler*/

unsigned short int
*RTCRead,*DA,*ADIn,*ADOut,*UART,*RTCOMMAND,*TSSEC,*TSMIN,*TSHOUR;
unsigned short int *TSDATE,*TSMONTH,*TSYEAR,*TSDOW,*RTINTREG;

unsigned short int TempIn[2880],
TempOut[2880],TimeYear[2880],TimeMonth[2880],TimeDate[2880];
unsigned short int TimeHour[2880],TimeMin[2880],TimeSec[2880],
InstVal[3], FirstLog;
unsigned char temperatur, InOut, RT_FLAG,UART_FLAG,
Nbr,ActFirstLog,ReadStart;
signed int j;
/*char ReadSignal1, ReadSignal2;

/* Initieringsprogram */

void init_RTC(void){
            /* Initiering av Real Time Clock */
      RTCOMMAND = (unsigned short int *) 0x0018011;
      /*Pekare*/
      TSSEC = (unsigned short int *) 0x0018003;
      TSMIN = (unsigned short int *) 0x0018002;
      TSHOUR = (unsigned short int *) 0x0018001;
      TSDATE = (unsigned short int *) 0x0018005;
      TSMONTH = (unsigned short int *) 0x0018004;
      TSYEAR = (unsigned short int *) 0x0018006;
      TSDOW = (unsigned short int *) 0x0018007;
      RTINTREG = (unsigned short int *) 0x0018010;
      *TSSEC = 0x00;          /*Starttid*/
      *TSMIN = 0x00;
      *TSHOUR = 0x00;
      *TSDATE = 0x00;
      *TSMONTH = 0x00;
      *TSYEAR = 0x00;
      *TSDOW = 0x00;
      *RTINTREG=0x08;    /*väljer periodisitet*/
      *RTCOMMAND = 0x1c; /*Inställning av klockfrek, int på av, mm*/
      return;
}

void init_UART(void){   /* Initiering av UART*/

      return;
}
```

```
 /* Diverse rutiner */
int Read_AD(char i){    /*Skickar omvandlingssignal till A/D*/
      if(i==1){
            *ADIn = 0;
            Nbr=2;
      }
      else{
            *ADOut = 0;
            Nbr=1;
      }
      return;
}

void wait(void){        /*Tidsfördröjningsfunktion*/
      int k;
      k=0;
      do{
            k++;
      }while(k<=0x5555);
}

void slask(void)        /*Program för brytpunkter exekvering*/
{
      return;
}

/********** Main program ////////////*/

      main(){
      /*Tilldelning av pekares värde*/
      RTCRead = (unsigned short int *) 0x0018000;
      DA = (unsigned short int *) 0x0030000;
      ADIn = (unsigned short int *) 0x0028000;
      ADOut = (unsigned short int *) 0x0028001;
      /*UART = (unsigned short int *) 0x0020000;*/

      /*Initiering*/
      init_RTC();
      init_UART();

      /*Här ligger huvudprogrammet*/
      _avben();               /*Tillåt avbrott*/
      Read_AD(1);
      j=-1;
      ActFirstLog=0;
      FirstLog=0;
      while(1){
            if(RT_FLAG==1 && UART_FLAG!=1){
      /*Sats som sparar undan värden vid avbrott från RT-klocka*/
                  j++;
/*Läser 1/100sec register för att latcha data till övriga
tidregister*/
                  ReadStart = *RTCRead;
                  TimeYear[j] = *TSYEAR;
                  TimeMonth[j] = *TSMONTH;
                  TimeDate[j] = *TSDATE;
                  TimeHour[j] = *TSHOUR;
                  TimeMin[j] = *TSMIN;
                  TimeSec[j] = *TSSEC;
                  /*Läser 1/100sec register för att avsluta läsning*/
```

```
                  ReadStart = *RTCRead;

                  /*Tilldelar värden från den temporära vektorn*/
                  TempIn[j] = InstVal[2];
                  TempOut[j] = InstVal[1];
                  slask();
                  /*Sätter ny startpunkt när vektorerna är fulla*/
                  if(j==2879){
                        j=-1;
                        FirstLog=0;
                        ActFirstLog=1;
                  }
                  if(ActFirstLog==1)
                        FirstLog=j+1;

                  /*Nollställer flagga från Avbrottsrutinen*/
                  RT_FLAG=0;
            }
            else if(UART_FLAG==1){
            /*Sats som skickar över data till PC*/
            }
      }
}


exp2(){     /* avbrottsprogram för RTC och AD */
      if(*RTINTREG > 0x7F){
            /*Sätter en flagga då ett interrupt fås från klockan*/
            RT_FLAG=1;
      }
      else{
            /*Sparar undan värde från bussen till temporär vektor*/
            /*OBS!!! Omvänd ordning på lagringsplats. Gör en ny
            läsning på AD'n på nästa kanal*/
            InstVal[Nbr]=*ADIn;
            Read_AD(Nbr);
      }
      return;
}

exp5(){
      /* avbrottsprogram för UART*/
      UART_FLAG=1;
      return;
}
```